CSS Futures

Web Development

CSS Futures

- CSS3
- CSS Preprocessors: SASS & LESS
- CSS Frameworks

CSS3

- CSS3 is the latest standard for CSS

- Combined with HTML5, CSS3 makes it possible to create highly interactive web sites, including games

- CSS3 is divided into modules, each addressing a specific aspect of CSS3, including selectors, box model, backgrounds and borders, text effects, 2D and 3D transformations, animations, multiple column layout, basic user interface, or media queries

- All of these modules go through a set of maturity levels, from announced (announced but no working draft or technical information is available yet) to recommendation (endorsed by W3C)

CSS3: Borders

- Custom borders and shadow effects without the need for a third-party image editing

```
.squareButton
{
    border:4px solid;
    border-color: blue;
    background: white;
    width:100px;
    height:40px;
}

.roundButton
{
    border:4px solid;
    border-radius: 25px;
    border-color: blue;
    background: white;
    width:100px;
    height:40px;
}
```

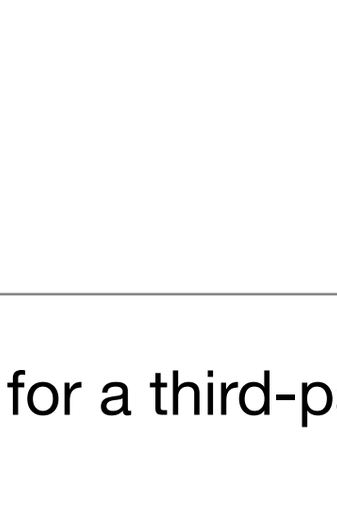
- The round border is created by defining the radius of the circle

CSS3: Shadows

- Custom borders and shadow effects without the need for a third-party image editing
- We can define the position of the shadow the blur distance, the spread and the color.

```
.shadow
{
    box-shadow:5px 5px 3px 2px #C1C1C1;
}

.squareButton
{
    border:4px solid;
    border-color: blue;
    background: white;
    width:100px;
    height:40px;
}
```



```
<div class = "shadow squareButton"> </div>
```

CSS3: Text Effects

- Text-shadow: shadows are applied to the text. Values specify the horizontal and vertical shadow, the blur distance, and the colour

```
#textShadow
{
     font-size: 40px;
     text-shadow: 10px 10px 10px blue;
     width:500px;
}
```

```
<div id = "textShadow"> This is a test width shadows</div>
```

This is a test width shadows

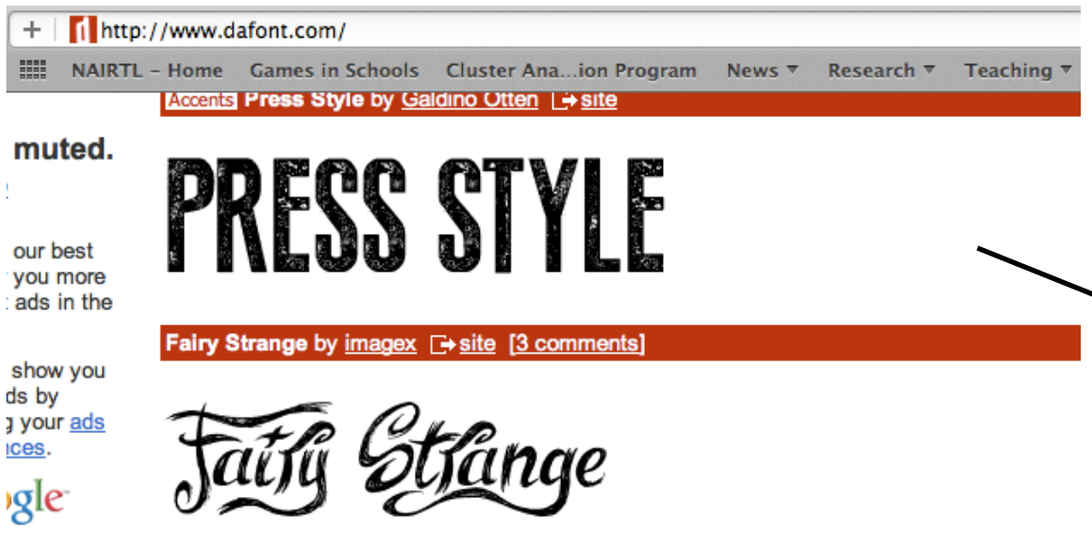
CSS3: Fonts

- CSS3 makes it possible to include fonts on your page that may not necessarily install on the clients browser

- With CSS3, the developer can include the necessary font on the server where the pages are hosted

- The process involved usually includes defining the font, accessing the font, and using the font in the CSS3 style sheet

CSS3: Fonts




```
@font-face
{
    font-family: lightHouse;
    src: url('lightHouse.ttf')
}
.newFont
{
    font-family: lightHouse;

}
```

```
<div class = "newFont"> Hello World </div>
```

Hello World

CSS3: Fonts

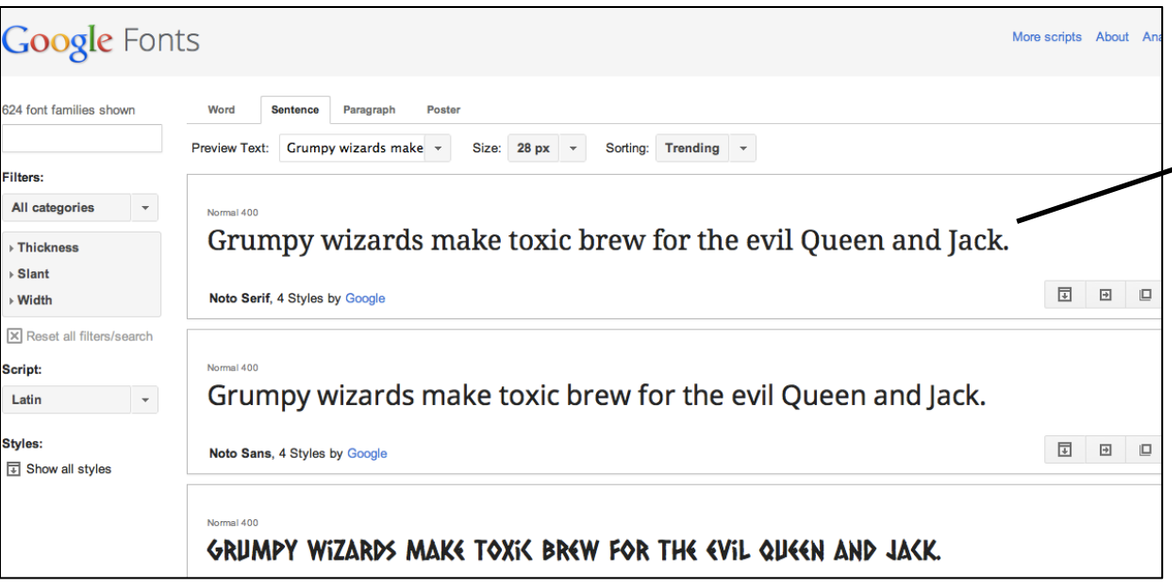
Fonts can also be originated from google fonts, which include a wide choice of fonts to be used on the web (http://www.google.com/fonts) and the process includes:

•Selecting your font, set the type of characters to be used,

•Adding automatic code to link to the corresponding style sheet

•Integrating the font to your CSS

CSS3: Fonts




```
<link rel="stylesheet" type="text/css" href="http://
fonts.googleapis.com/css?family=Tangerine">
```

```
.tangerineFont
{
    font-family: 'Tangerine', serif;
    font-size:30px;

}
```

```
<div class = "tangerineFont"> Hello World with
Tangerine</div>
```

Hello World with Tangerine

CSS3: Transforms

- CSS3 adds the ability to apply transformations to elements, including rotating, or scaling. A node can be called a descendant node if it's a child, grandchild, and so on, of another node.

- Although this functionality is implemented for most popular browsers, some browser-specific definitions need to be applied

- While the statement *transform:* is used to apply a transformation (as per W3c guidelines), a prefix needs to be added for some browsers

- These prefixes indicate a browser-specific implementation of a CSS feature and it is good practice to include all of these in the CSS style sheet so that the page is compatible with all browsers.

CSS3: Transforms

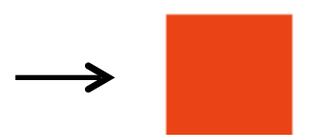
- For Firefox: use the prefix –moz-
- For Internet Explorer: use the prefix –ms-
- For chrome and safari: use the prefix –webkit-
- For Opera: use the prefix –o-

```
.box
{
    color: red;
    background: red;
    width: 50px;
    height: 50px;
    margin:20px;



}
.rotate45
{
    transform: rotate(45deg);
    -ms-transform: rotate(45deg);
    -webkit-transform: rotate(45deg);
    -o-transform: rotate(45deg);
    -moz-transform: rotate(45deg);
}
```

```
<div class ="box">
```



```
<div class ="box rotate45">
```

CSS3: Transitions

- Transitions make it possible to apply smooth transitions to an element while between two styles

- When the transition applies to several parameters, it is also possible to specify how each individual parameter will evolve overtime

1

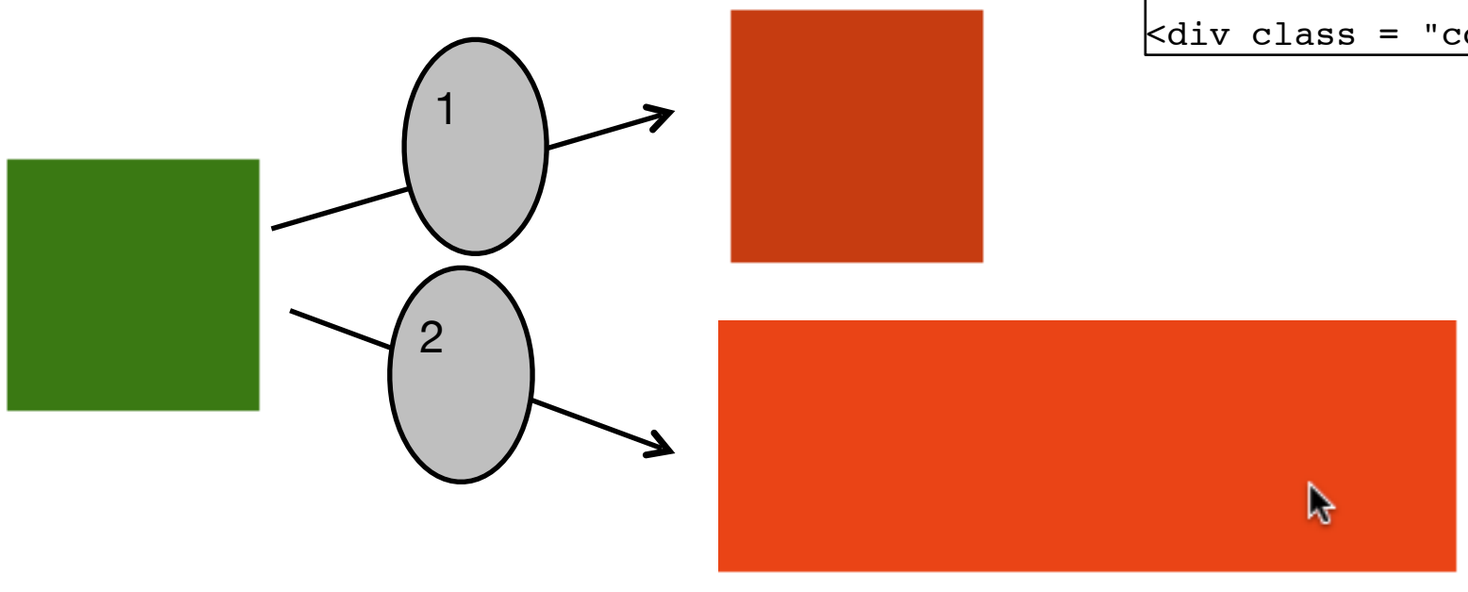
```
.color_transition
{
    background: green;
    width: 100px;
    height: 100px;
    -moz-transition: 2s;
}

.color_transition:hover
{
    background:red;
}
```

2

```
.color_transition:hover
{
    background:red;
    width: 300px;
}
```

```
<div class = "color_transition"></div>
```



CSS3: Transitions

- We can also specify how long the transition for each parameter will last

```
.color_transition
{
    background: green;
    width: 100px;
    height: 100px;
    -moz-transition: background 2s, width .5s;
}

.color_transition:hover
{
    background:red;
}
```

```
.color_transition:hover
{
    background:red;
    width: 300px;
}
```

```
<DIV class = "color_transition"></DIV>
```

CSS3: Animations

- CSS3 makes it possible to create animations, in a similar manner to Flash or video editing using keyframes

- An animation is defined and applied to a specific element

- The duration of the animation is defined along with key moments (keyframes) and the values of some parameters at these moments

- An gradual change between key values creates a smooth animation

```
.animatedBox
{
    width:100px;
    height:100px;
    background:red;
    -moz-animation:myAnimation 4s;
}
@-moz-keyframes myAnimation /* Firefox */
{
    0%   {width:0px; height:0px}
    25%  {width:100px;height:100px}
    50%  {width:0px;height:0px}
    100% {width:100px;height:100px}
}
```

```
<DIV class ="box rotate45 animatedBox">
```

CSS3: Animations

- Additional attributes for an animation include:

- Animation-iteration-count: how many times the animation should be repeated (number of times to be repeated or *infinite* to repeat the animation indefinitely)

- Animation-play-state: whether the animation is *running* or *paused*

```
.animatedBox
{
    width:100px;
    height:100px;
    background:red;
    -moz-animation:myAnimation;
        animation-duration: 5s;
        animation-iteration-count: infinite;
        animation-play-state: running;
}
@-moz-keyframes myAnimation /* Firefox */
{
    0%   {width:0px; height:0px}
    25%  {width:100px;height:100px}
    50%  {width:0px;height:0px}
    100% {width:100px;height:100px}
}
```

```
<DIV class ="box rotate45 animatedBox">
```

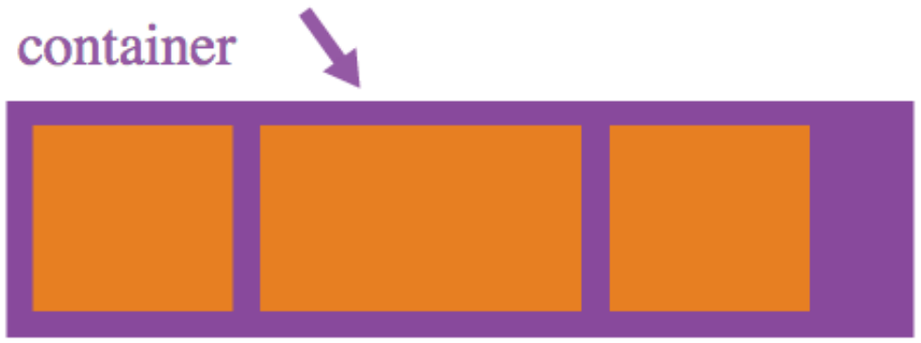
Flexbox

- The Flexbox Layout (Flexible Box) module (currently a W3C Last Call Working Draft) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word "flex").

- Give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes).

- A flex container expands items to fill available free space, or shrinks them to prevent overflow.

https://css-tricks.com/snippets/css/a-guide-to-flexbox/



Properties for the Parent

(flex container)

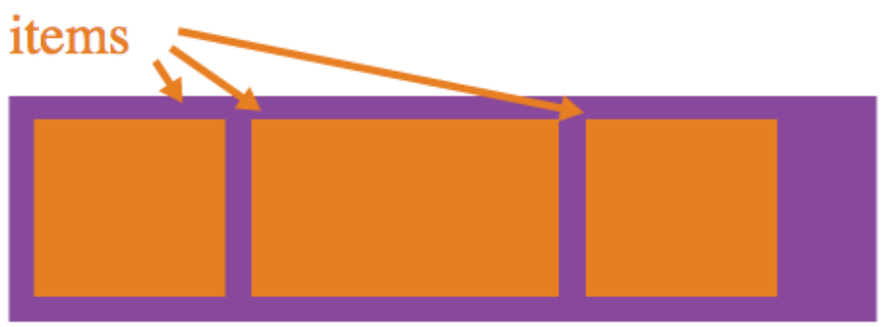
display

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

CSS

```css
.container {
  display: flex; /* or inline-flex */
}
```

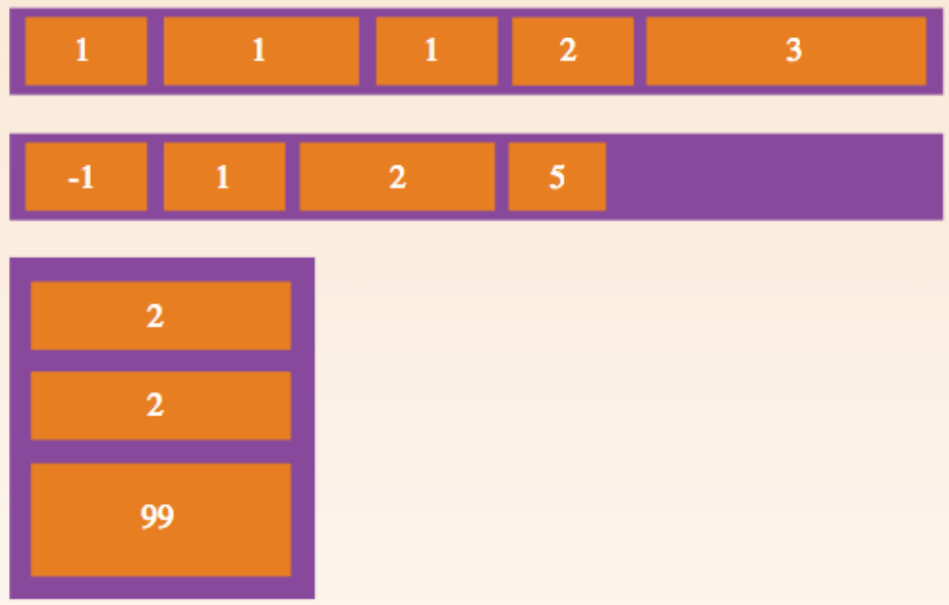
Note that CSS columns have no effect on a flex container.



Properties for the Children

(flex items)

order



By default, flex items are laid out in the source order. However, the `order` property controls the order in which they appear in the flex container.

CSS

```css
.item {
  order: <integer>;
}
```

flex-direction

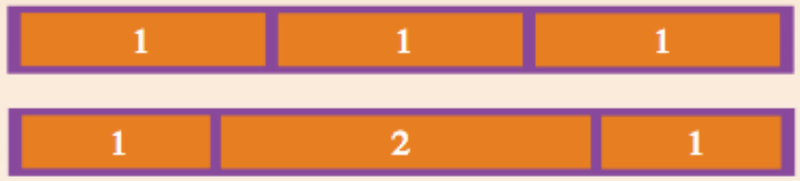
This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

CSS

```
.container {
  flex-direction: row | row-reverse | column | columr
}
```

- `row` (default): left to right in `ltr` ; right to left in `rtl`
- `row-reverse` : right to left in `ltr` ; left to right in `rtl`
- `column` : same as `row` but top to bottom
- `column-reverse` : same as `row-reverse` but bottom to top

flex-grow



This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have `flex-grow` set to 1, every child will set to an equal size inside the container. If you were to give one of the children a value of 2, that child would take up twice as much space as the others.

CSS

```
.item {
  flex-grow: <number>; /* default 0 */
}
```

Negative numbers are invalid.

flex-shrink

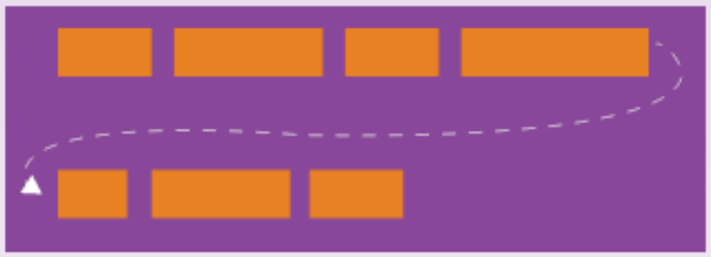
This defines the ability for a flex item to shrink if necessary.

CSS

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

Negative numbers are invalid.

flex-wrap



By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property. Direction also plays a role here, determining the direction new lines are stacked in.

CSS

```
.container{
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

- `nowrap` (default): single-line / left to right in `ltr`; right to left in `rtl`
- `wrap`: multi-line / left to right in `ltr`; right to left in `rtl`
- `wrap-reverse`: multi-line / right to left in `ltr`; left to right in `rtl`

flex-basis

This defines the default size of an element before the remaining space is distributed. The `main-size` value makes it match the `width` or `height`, depending on which is relevant based on the `flex-direction`.

CSS

```
.item {
  flex-basis: <length> | auto; /* default auto */
}
```

If set to `0`, the extra space around content isn't factored in. If set to `auto`, the extra space is distributed based on it's flex-grow value.
See this graphic.

CSS Preprocessing languages

- A CSS preprocessor helps write maintainable, future-proof code and it will seriously reduce the amount of CSS you have to write.

- Where these tools shine best are in large-scale user interfaces that require huge stylesheets and many style rules.

- Two candidates:

  - SASS

  - LESS

SASS

INSTALL LEARN SASS BLOG DOCUMENTATION GET INVOLVED LIBSASS

CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

Operators

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

Operators

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

Operators

```scss
// _reset.scss

html,
body,
ul,
ol {
   margin: 0;
  padding: 0;
}
```

```scss
/* base.scss */

@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

Operators

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

Operators

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}

.warning {
  @extend .message;
  border-color: yellow;
}
```

Preprocessing

Variables

Nesting

Partials

Import

Mixins

Inheritance

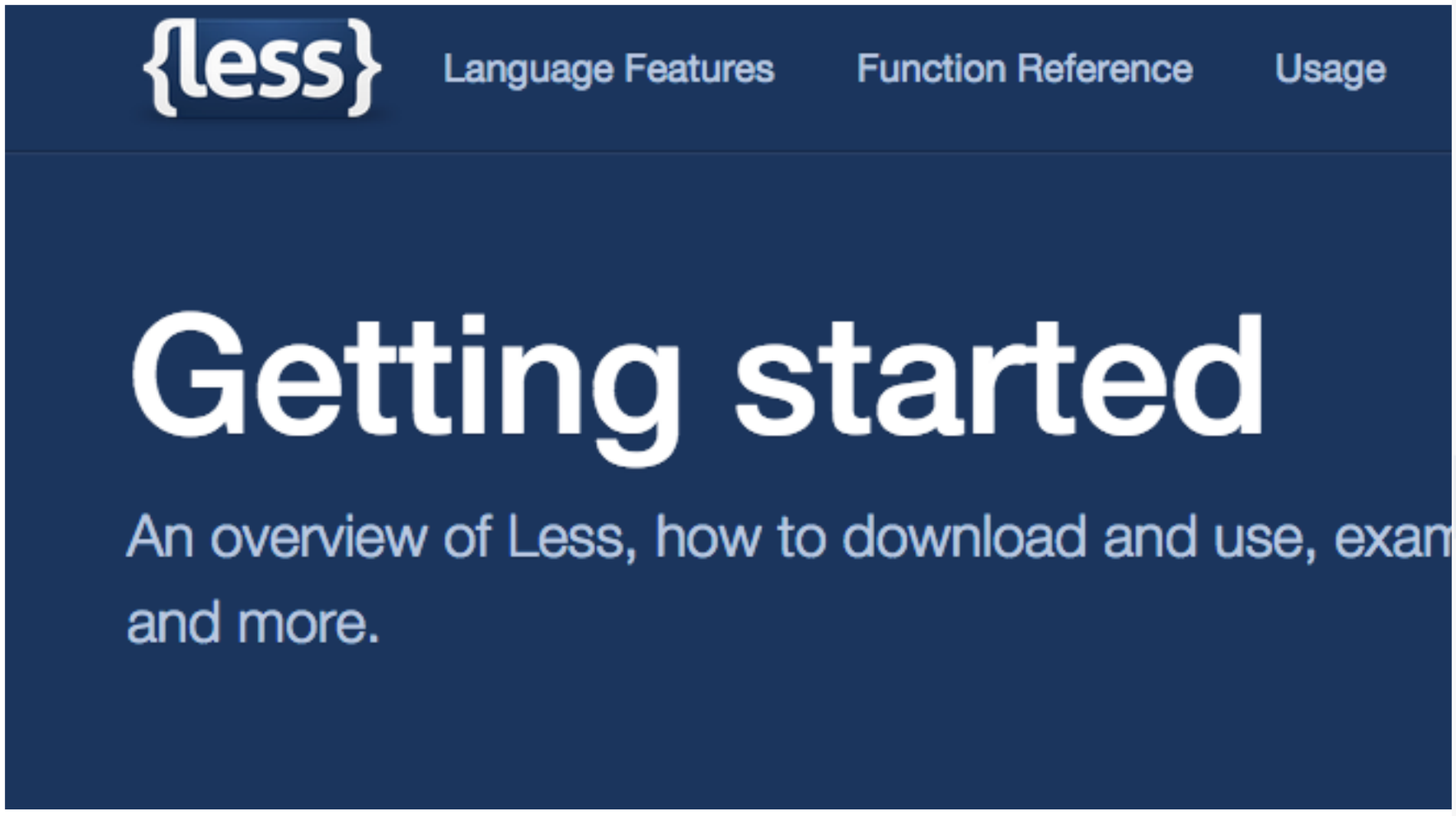
Operators

```
.container { width: 100%; }

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complimentary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

LESS



- Less is a CSS pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques that allow you to make CSS that is more maintainable, themable and extendable.

CSS frameworks

From Wikipedia, the free encyclopedia

CSS frameworks are pre-prepared software frameworks that are meant to allow for easier, more standards-compliant web design using the Cascading Style Sheets language. Most of these frameworks contain at least a grid. More functional frameworks also come with more features and additional JavaScript based functions, but are mostly design oriented and unobtrusive. This differentiates these from functional and full JavaScript frameworks.

Some notable and widely used examples are Bootstrap or Foundation.

CSS frameworks offer different modules and tools:

- Reset-Stylesheet
- grid especially for responsive web design
- web typography
- set of icons in sprites or iconfonts
- styling for tooltips, buttons, elements of forms
- parts of graphical user interfaces like Accordion, tabs, slideshow or Modal windows (Lightbox)
- Equalizer to create equal height content
- often used css helper classes (*left*, *hide*)

Bigger frameworks use CSS interpreter like LESS or SASS.

CSS Frameworks

Semantic UI

MIT

> UI is the vocabulary of the web.

Semantic empowers designers and developers by creating a language for sharing UI.
Lose the Hieroglyphics: Semantic is structured around natural language conventions to make development more intuitive.
Have a conversation with your components: Semantic gives you a variety of UI components with real-time debug output, letting your code tell you what its doing.

Site

Github

Bootstrap

MIT

> Sleek, intuitive, and powerful front-end framework for faster and easier web development.

Site

Github

Foundation

MIT

> The most advanced responsive front-end framework in the world.

Foundation 3 is built with Sass, a powerful CSS preprocessor, which allows us to much more quickly develop Foundation itself and gives you new tools to quickly customize and build on top of Foundation.

Site

Github

UIkit

MIT

> A lightweight and modular front-end framework for developing fast and powerful web interfaces.

UIkit gives you a comprehensive collection of HTML, CSS, and JS components. It can be extended with themes and is easy to customize to create your own look.

Site

Github